

### **Remarks**

Claims 1-52 are pending in this application. The examiner has rejected claims 1-52 under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,872,973 to Mitchell. Applicants have amended claims 1, 19, 36, and 40.

#### **A. Standard for Rejection Under 35 U.S.C. § 102**

“A claim is not anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference.” *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631 (Fed. Cir. 1987). “The identical invention must be shown in complete detail as is contained in the . . . claim.” *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1336 (Fed. Cir. 1989). Thus, a rejection of any one of the pending claims is not permissible unless each element of the rejected claim are present in Mitchell. Here, Mitchell does not disclose each element of any of the rejected claims.

#### **B. Mitchell Does Not Disclose the Claimed Invention**

A plain reading of Mitchell demonstrates that Mitchell does not disclose the claimed invention. The claimed invention of the present application is directed to an object-oriented programming environment that includes a shared, distributed programming environment in which each object of the environment is able to map commands directed to the object to one of several behaviors of the object on the basis of a mapping function that is included within the object. The placement of the mapping functionality of the invention within the object allows the object to be autonomous and allows the object to function without regard to the location of the object within the programming environment. In addition, because the mapping functionality of the object is intrinsic to and within the object, an object can function in the programming environment without the necessity of defining relationships between the object

and other objects of the programming environment.

In direct contrast, Mitchell is directed to a programming environment in which a “semantic link” — also referred to in Mitchell as a “surrogate object” or “mapper” — is placed *between* two other objects and serves to communicate with each of the objects. Mitchell provides that:

[A] semantic link in the present invention is a system that synchronizes two server objects by creating a generic client between the servers.

...

The main function of the invention is to specify and then instantiate semantic links between objects that have been defined in a dynamic object-oriented language.

...

A semantic link is created through the instantiation of a surrogate object, called a mapper, that uses probing and dynamic binding to attach to both of its patron objects. The mapper is the client and both patron objects are servers to the mapper.

...

The objects are synchronized (that is a “message” is passed from one object to the other) through the mapper when the attribute changes its value, and the probes are called back.

...

In other words, two objects can be attached to each other by a third object without the first two objects becoming clients, thus creating a server/server architecture between the two objects.

(Mitchell, column 7, lines 36-38, 46-48, and 58-62; and column, 8, lines 18-21 and 32-35). The placement of a mapping function between two objects is also plainly shown in the figures of Mitchell. Figure 1 of Mitchell, for example, depicts a semantic link or mapper in the form of EosMapFieldToField object 103, which is positioned communicatively between two other

objects, which are identified as patron object 102 and patron object 106. It is plain from Mitchell that Mitchell only discloses a mapping function placed between — *not within* — two objects.

Mitchell does not disclose the claimed invention of a shared, object-oriented environment in which the mapping of commands to behaviors for an object is accomplished according to a mapping functionality that resides within the object itself. In addition, independent claims 1, 19, 36, and 40 have been amended herein to clarify that the mapping function of the invention is within each object, thereby allowing each object to function as an autonomous unit such that the object can be moved within the computer systems of the shared environment and function independently of its location in the shared environment. Further, independent claims 1, 19, 36, and 40 have been amended to clarify that the existence of the mapping function within each object allows the objects to function without the necessity of defining relationships between the object and other objects in the programming environment. Each claim of the pending application includes the recitation of claim elements that are not disclosed or suggested by Mitchell.

1. *Independent Claim 1*

Claim 1 is directed to a storage medium containing software for manipulating computer-implemented objects in a computer system. Claim 1 includes the recitation of code to create an object. The claimed object of claim 1 includes “mapping logic able to map a command received at the receiver logic, on the basis of a characteristic of the command, to a selected behavior logic for execution of the selected behavior logic.” This element of the claimed object of claim 1 is simply not present in Mitchell, as Mitchell is directed to the placement of the mapping function *between* two objects. In contrast, the mapping function of the object of claim 1 is within the object itself.

The examiner has pointed to column 12, lines 18-48 of Mitchell as disclosing the mapping of commands to behaviors. The discussion contained in column 12, lines 18-48 of Mitchell is not directed to a command-behavior mapping function that is included within the target object. The mapping that is being described in this section is the mapping of one object to another object through an external mapping function. The “mapping dialog” that is discussed in this section is a “dialog box” for establishing the external mapping function between two objects. An example of this dialog box is shown in Figure 4 of Mitchell, which plainly depicts an external mapping function between “ObjectA” and “ObjectD.”

In plain contrast, the object of the present invention includes a mapping function within the object itself. Claim 1 has been amended to clarify that the placement of the mapping function within the object allows the object to be autonomous and further allows the object to function within the shared environment without reference to the location of the object in the shared environment and without the necessity of defining relationships between the object and other objects. The inclusion of the mapping function within the object is simply not disclosed or suggested in Mitchell. Applicants submit that the rejection of claim 1 over Mitchell should be withdrawn.

## *2. Independent Claim 19*

Independent claim 19 is directed to a method for manipulating a computer-implemented object in a distributed system. The step of “creating a plurality of objects” of claim 19 includes the steps of:

selecting a behavior logic of the set of behavior logics corresponding to the command on the basis of a mapping logic within the object that maps commands to behavior logics of the set of behavior logics on the basis of a characteristic of the command;

(Claim 19). This step is not disclosed in Mitchell, as Mitchell is directed to the placement of the mapping function *between* two objects. In direct contrast with Mitchell, the mapping function of the claimed object is “within the object,” thereby allowing the object to be autonomous and operable without regard to its location.

As discussed with respect to claim 1 above, the examiner has pointed to column 12, lines 18-48 of Mitchell as disclosing the mapping of commands to behaviors. The discussion contained in column 12, lines 18-48 of Mitchell is not directed to a command-behavior mapping function that is included within the target object. The mapping that is being described in this section is the mapping of one object to another object through an external mapping function. The “mapping dialog” that is discussed in this section is a “dialog box” for establishing the external mapping function between two objects. An example of this dialog box is shown in Figure 4 of Mitchell, which plainly depicts an external mapping function between “ObjectA” and “ObjectD.”

In plain contrast to this passage of Mitchell, the object of claim 19 includes a mapping function within the object itself. Claim 19 has been amended to clarify that the placement of the mapping function within the object allows the object to be autonomous and further allows the object to function within the shared environment without reference to the location of the object in the shared environment and without the necessity of defining relationships between the object and other objects. The inclusion of the mapping function within the object is simply not disclosed or suggested in Mitchell. Applicants submit that the rejection of claim 19 over Mitchell should be withdrawn.

### 3. *Independent Claim 36*

Independent claim 36 is directed to a method for designing an application for execution on at least one computer system. A step of “manipulating the object” of claim 36 includes the step of:

mapping members of a first set of commands to members of the set of behavior logics, wherein the mapping function of an object is included within the object;

...

configuring a receiver logic to receive a command and initiate the execution of a behavior logic corresponding to the command in response to the mapping of the command to the behavior logic.

(Claim 36). These steps are not disclosed in Mitchell, as Mitchell is directed to the placement of the mapping function *between* two objects. In direct contrast with Mitchell, the mapping function of the claimed object is included “within the object,” thereby allowing the object to be autonomous and operable without regard to its location and without the necessity of defining relationships between the object and other objects.

As discussed with respect to claims 1 and 19 above, the examiner has pointed to column 12, lines 18-48 of Mitchell as disclosing the mapping of commands to behaviors. The discussion contained in column 12, lines 18-48 of Mitchell is not directed to a command-behavior mapping function that is included within the target object. The mapping that is being described in this section is the mapping of one object to another object through an external mapping function. The “mapping dialog” that is discussed in this section is a “dialog box” for establishing the external mapping function between two objects. An example of this dialog box is shown in Figure 4 of Mitchell, which plainly depicts an external mapping function between “ObjectA” and “ObjectD.”

In plain contrast to this passage of Mitchell, the objects of claim 36 include a mapping function with each object itself. Claim 36 has been amended to clarify that the placement of the mapping function within each object allows the object to be autonomous and further allows the object to function within the shared environment without reference to the location of the object in the shared environment and without the necessity of defining relationships between the object and other objects. The inclusion of the mapping function within the object is simply not disclosed or suggested in Mitchell. Applicants submit that the rejection of claim 36 over Mitchell should be withdrawn.

4. *Independent Claim 40*

Independent claim 40 is directed to a processor-based system. The system includes code to create an object, and the created object includes:

a mapping logic able to map a command received at the receiver logic to a selected behavior logic for execution of the selected behavior logic on the basis of a characteristic of the command.

(Claim 40). An object for use in a shared environment that includes a set of behavior logics and a mapping logic within the object is not disclosed in Mitchell. Mitchell, in contrast, is directed to the placement of the mapping function *between* two objects. In direct contrast with Mitchell, the mapping function of the claimed object of claim 40 is included within the object, thereby allowing the object to be autonomous and operable without regard to its location and without the necessity of defining relationships between the object and other objects.

As discussed with respect to claim 1, 19, and 36 above, the examiner has pointed to column 12, lines 18-48 of Mitchell as disclosing the mapping of commands to behaviors. The discussion contained in column 12, lines 18-48 of Mitchell is not directed to a command-behavior mapping function that is included within the target object. The mapping that

is being described in this section is the mapping of one object to another object through an external mapping function. The “mapping dialog” that is discussed in this section is a “dialog box” for establishing the external mapping function between two objects. An example of this dialog box is shown in Figure 4 of Mitchell, which plainly depicts an external mapping function between “ObjectA” and “ObjectD.”

In plain contrast to this passage of Mitchell, the object of claim 40 includes a mapping logic within the object itself. Claim 40 has been amended herein to clarify that the placement of the mapping logic within the object allows the object to be autonomous and further allows the object to function within the shared environment without reference to the location of the object in the shared environment and without the necessity of defining relationships between the object and other objects. The inclusion of the mapping function within the object is simply not disclosed or suggested in Mitchell. Applicants submit that the rejection of claim 40 over Mitchell should be withdrawn.

#### 5. *Independent Claim 46*

Independent claim 46 is directed to a software architecture. Claim 46 includes the recitation of a distributed system that includes “a plurality of shared environments.” There is no disclosure in Mitchell of multiple shared environments. For the disclosure of a software architecture that includes multiple shared environments, the examiner points to three lines in column 6, two lines in the abstract, and Figures 1-3 of Mitchell. None of these passages or the figures discloses multiple shared environments. Instead, these passages simply disclose linking of objects and the external mapping of objects. There is not even a suggestion in any of these passages of a software architecture that includes *multiple* shared environments.

In addition, claim 46 includes “a Kernel subclass of the CommandReceiver class” that includes “code to instantiate objects of the CommandReceiver class” and “code to destroy objects of the CommandReceiver class.” Neither of these code elements is disclosed in Mitchell. First, with respect to the element of “code to instantiate objects of the CommandReceiver class,” the examiner has pointed to column 29, lines 55-67 of Mitchell. This passage of Mitchell says nothing about the instantiation of objects. Rather, this passage speaks again to links between objects. There is absolutely no mention in this passage of code for instantiating, or creating, an object of the CommandReceiver class.

In addition, the claim element of a Kernel subclass that includes “code to destroy objects of the CommandReceiver class,” is not disclosed in Mitchell. For this element, the examiner points to the “garbage collection” discussion in column 17, lines 29-48 of Mitchell. There is no disclosure in this passage of code within a Kernel subclass of the CommandReceiver class for destroying objects of the CommandReceiver class. Code of this sort is simply not disclosed in Mitchell. Applicants respectfully submit that the rejection of claim 46 should be withdrawn.

In sum, each of the pending claims includes claim elements that are not present in Mitchell. Because Mitchell does not disclose each element of the claimed invention, a rejection of the pending claims under 35 U.S.C. § 102 over Mitchell is improper. The rejection of the pending claims on the basis of Mitchell should be withdrawn.

**C      Dependent Claims 2-18, 20-35, 37-39, 41-45, and 47-52**

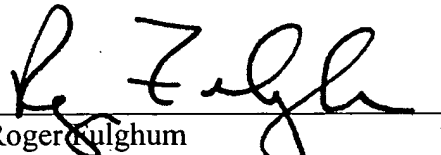
Dependent claims 2-18, 20-35, 37-39, 41-45, and 47-52 will not be discussed individually herein, as each depends from an otherwise allowable base claim. Applicants submit that the rejection of claims 2-18, 20-35, 37-39, 41-45, and 47-52 should be withdrawn and these

claims should be passed to issuance.

**Conclusion**

Applicants respectfully submit that the rejection of claims 1-52 should be withdrawn, and these claims should be passed to issuance.

Respectfully submitted,

A handwritten signature in black ink, appearing to read 'R. Gulghum', written over a horizontal line.

Roger Gulghum  
Registration No. 39,678

Baker Botts L.L.P.  
910 Louisiana  
One Shell Plaza  
Houston, Texas 77002-4995  
(713) 229-1707

Baker Botts Docket Number: 068820.0102

Date: December 13, 2005